Homework 9: Barriers

제출 마감: 2018. 11. 29. 11:00 am

제출처: 오준택 조교 (na94jun@gmail.com)

본 과제는 `xv6`와 `qemu`를 사용하지 않고, 자신의 노트북 혹은 데스크톱에서 진행한다.

먼저, `barrier.c` 파일을 다운받고 컴파일하도록 한다. 컴파일 명령은 다음과 같다.

```
$ gcc -g -O2 -pthread barrier.c
$ ./a.out 2
Assertion failed: (i == t), function thread, file barrier.c, line 55.
```

프로그램 실행에 인자로 전달된 2는 실행되는 쓰레드의 개수이다. 각 쓰레드들은 루프문을 실행한다. 각 루프에서는 `barrier()`와 `usleep()`이 순서대로 호출되는데, `usleep()`의 인자로 난수가 전달되어, 쓰레드는 랜덤한 시간동안 sleep한다. 현재 `barrier.c`를 실행하면 assert가 발생할 것이다. 이는 한 쓰레드가 `barrier()`를 호출하기 전에 다른 한 쓰레드가 `barrier()`를 반환하기 때문이다. 본 과제의 목표는 쓰레드가 `barrier()`를 호출하면 모든 쓰레드가 `barrier()`를 호출할 때까지 대기하도록 `barrier()`를 구현하는 것이다.

지난 과제였던 Homework 6에서 lock primitives를 사용했다. 본 과제를 수행하기 위해서, 새로운 lock primitive인 컨디션 변수를 참고해야 할 것이다. 새로운 lock primitives는 다음과 같다.

```
pthread_cond_wait(&cond, &mutex);  // 뮤텍스를 해제하고, cond에 대해서 sleep
pthread_cond_broadcast(&cond);     // cond에 대해 sleep한 모든 쓰레드를 깨움
```

`pthread_cond_wait`은 내부에서 sleep을 하는데, sleep을 하기 전에 mutex를 해제하고, sleep에서 깨어나서 return 하기 전에 mutex를 다시 잡는다.

`barrier.c`에는 `struct barrier`, `barrier_init()`, `barrier()`가 구현되어 있다. `struct barrier`와 `barrier_init()`에는 이미 컨디션 변수 관련 코드가 추가되어 있다. 해당 코드를 참조하여 panic이 발생하지 않도록 `barrier()`를 완성시키시오.

참고 사항이 두가지 있다.

- `barrier()`는 연속적으로 호출될 것이고, 각 쓰레드는 sleep하고 깨어나고를 반복할 것이다. 이 때, 쓰레드들이 깨어날 때까지를 한 round라고 부를 것이다. `bstate.round`에는 현재 round가 저장될 것이다. 즉, 모든 쓰레드가 barrier()를 호출하여 쓰레드들을 깨울 때, `bstate.round`가 1 증가해야 한다.
- 다른 쓰레드들이 `barrier()`를 반환하기 전에 한 쓰레드가 loop를 실행할 수 있다. 매 round마다 `bstate.nthread`를 재사용할 것이다. 주의해야 할 점은 `barrier()`를 종료하고 다음 loop를 실행하려는 쓰레드가 이전 round에서 사용중인 `bstate.nthread`를 증가시키지 않도록 하는 것이다.

쓰레드 개수를 1개, 2개, 그 이상으로 늘려가면서 잘 실행되는지 확인하시오.

Submit: 수정한 barrier.c 파일

제출 양식: hw9_학번.S